

CAS FEE - Serverless Data Management

React und Firebase

Mirko Stocker – mirko.stocker@ost.ch



CAS FEE - Serverless Data Management

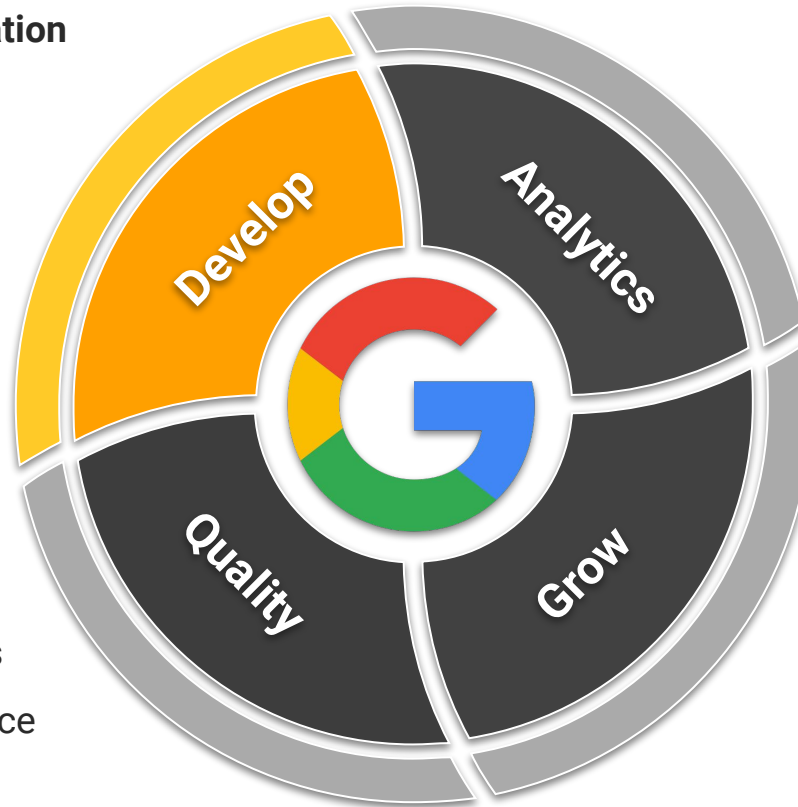
Google Firebase



Firebase – Backend Services für Mobile und Web

- **Authentication**
- **Database**
- Storage
- **Hosting**
- **Functions**
- ML Kit

- Crashlytics
- Performance
- Test Lab



- Events
- Conversions
- Audiences
- Funnels
- ...

- A/B Testing
- Cloud Messaging
- Remote Config
- Dynamic Links
- ...

Kosten

- Firebase läuft in der Google Cloud Platform
- Gratispläne für kleinere Projekte
- Kosten skalieren mit Nutzung

Products	Free Spark Plan Generous limits to get started	Pay as you go Blaze Plan Calculate pricing for apps at scale ✓ Free usage from Spark plan included*
Authentication Phone Auth - US, Canada, and India [?] Phone Auth - All other countries [?] Other Authentication services	10k/month 10k/month ✓	\$0.01/verification \$0.06/verification ✓
Cloud Firestore Stored data Network egress Document writes Document reads Document deletes	1 GiB total 10GiB/month 20K/day 50K/day 20K/day	\$0.18/GiB Google Cloud pricing \$0.18/100K \$0.06/100K \$0.02/100K
Cloud Functions [?] Invocations GB-seconds CPU-seconds Outbound networking	125K/month 40K/month 40K/month Google services only	\$0.40/million \$0.0025/thousand \$0.01/thousand \$0.12/GB
Hosting GB stored GB transferred Custom domain & SSL Multiple sites per project	1 GB 10 GB/month ✓ ✓	\$0.026/GB \$0.15/GB ✓ ✓

Plattformen

- Hauptfokus von Firebase sind Mobile- und Web-Apps



Get started for Android

API Reference

Codelabs



Get started for iOS

API Reference

Codelabs



Get started for Web

API Reference

Codelabs



Get started for C++

API Reference



Get started for Unity

API Reference



Get started for Admin

API Reference

- Verschiedenste Programmiersprachen werden unterstützt

iOS



NODE

JAVA















GO



Firebase Authentication

- Verschiedenste Authentifizierungsmöglichkeiten
- Backend Services für Authentifizierung und einfache Userverwaltung
 - 👏 Passwort zurücksetzen
 - 👏 Accounts sperren
- SDKs für diverse Plattformen
- Vorgefertigte UI Libraries

Provider	Status
 Email/Password	Enabled
 Phone	Disabled
 Google	Enabled
 Play Games	Disabled
 Game Center Beta	Disabled
 Facebook	Disabled
 Twitter	Disabled
 GitHub	Disabled
 Yahoo	Disabled
 Microsoft	Disabled
 Apple	Disabled
 Anonymous	Disabled

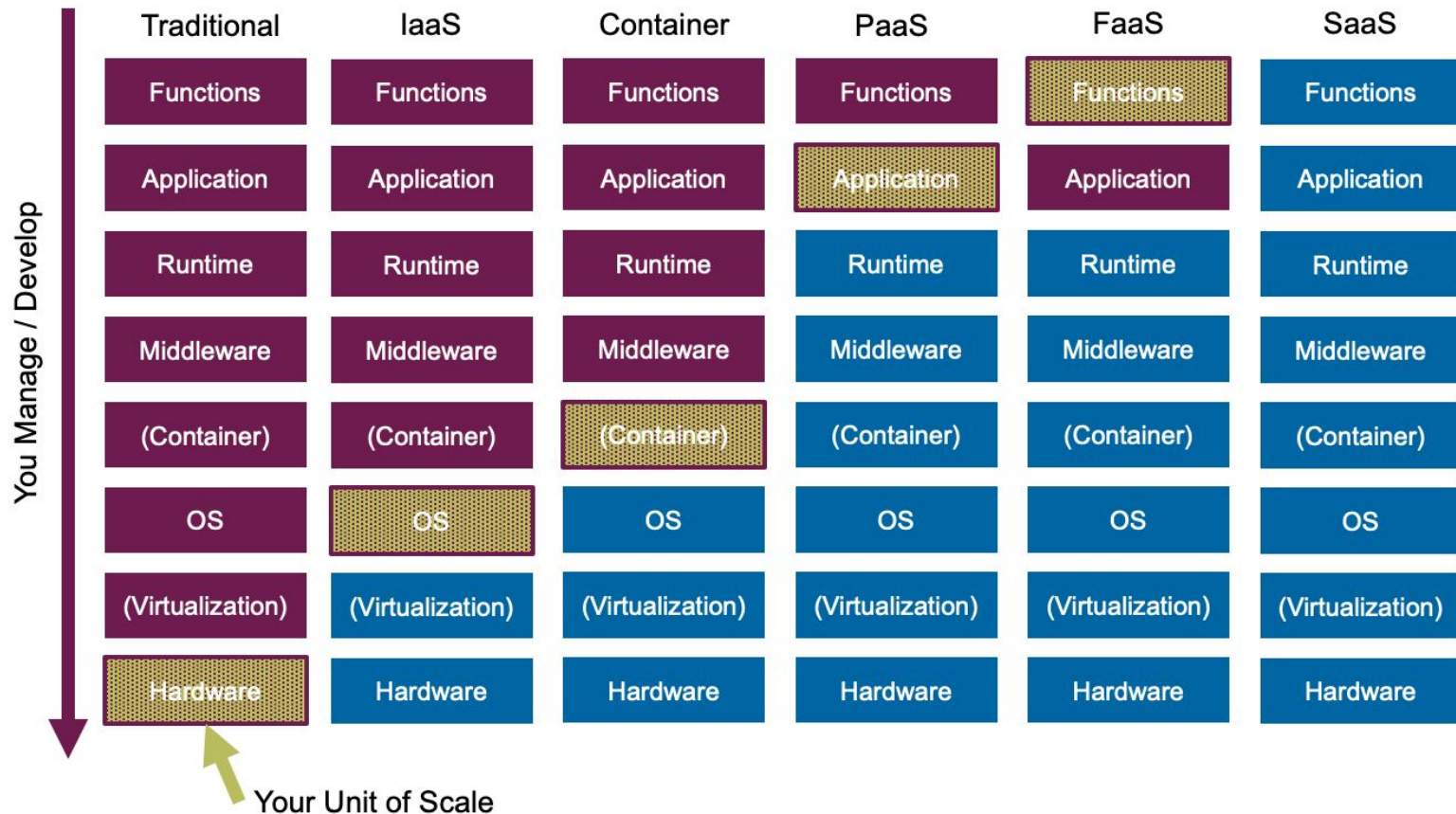


Firebase Hosting

- Einfaches Hosting für statischen Content
 - 👍 Wird immer per HTTPS ausgeliefert
 - 👍 Automatisches Caching in CDNs
 - 👍 Eigene Domains möglich
 - 👍 Rollbacks auf alte Versionen
- “Dynamischer” Content nur über **Cloud Functions**, wenn das nicht reicht:
 - 👉 PaaS: Google App Engine
 - 👉 Docker: Google Container oder Kubernetes Engine
- Für Hosting eine Alternative zu Amazons S3



Firebase Cloud Functions



Serverless Computing

■ Cloud Provider verwaltet Functions:

- 👉 Deployment geschieht on-demand
- 👉 Plattform bestimmt die Parallelisierung
- 👉 Entwickler hat keine Kontrolle über laufende Instanzen
- 👉 Funktionen sind Stateless, d.h. kein Server Session State
- 👉 Abgerechnet werden Aufrufe und Laufzeit der Funktion

■ Limitationen

- 👎 Ausführungszeit und Memory sind vom Provider begrenzt
- 👎 Teilweise hohe Latenz (Code muss erst deployed werden!)



Cloud Functions



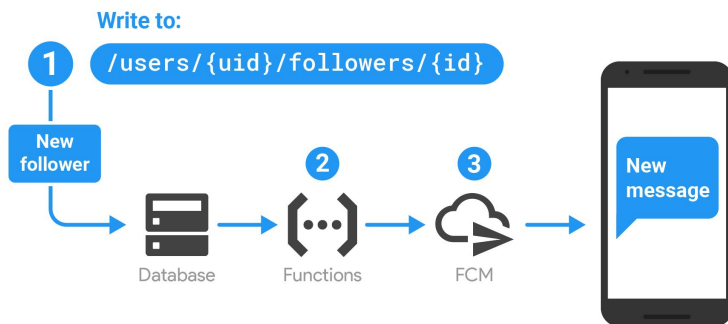
Invocations	\$0.40/million
GB-seconds	\$0.0025/thousand
CPU-seconds	\$0.01/thousand
Outbound networking	\$0.12/GB



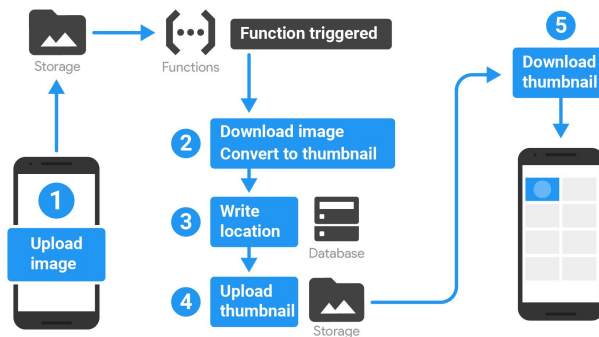
Firebase Cloud Functions

■ Typische Anwendungsszenarien

- ☁ Code als Reaktion auf einen Event ausführen
- ☁ Administration (Cron Jobs)
- ☁ REST API für Mobile und SPAs zur Verfügung stellen



<https://firebase.google.com/docs/functions/use-cases>





Cloud Firestore





Cloud Firestore vs Realtime Database

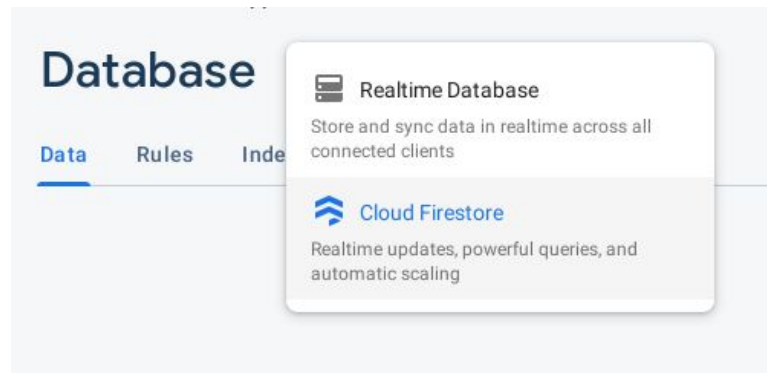
- Firebase hat zwei Datenbankprodukte



Realtime Database



Cloud Firestore

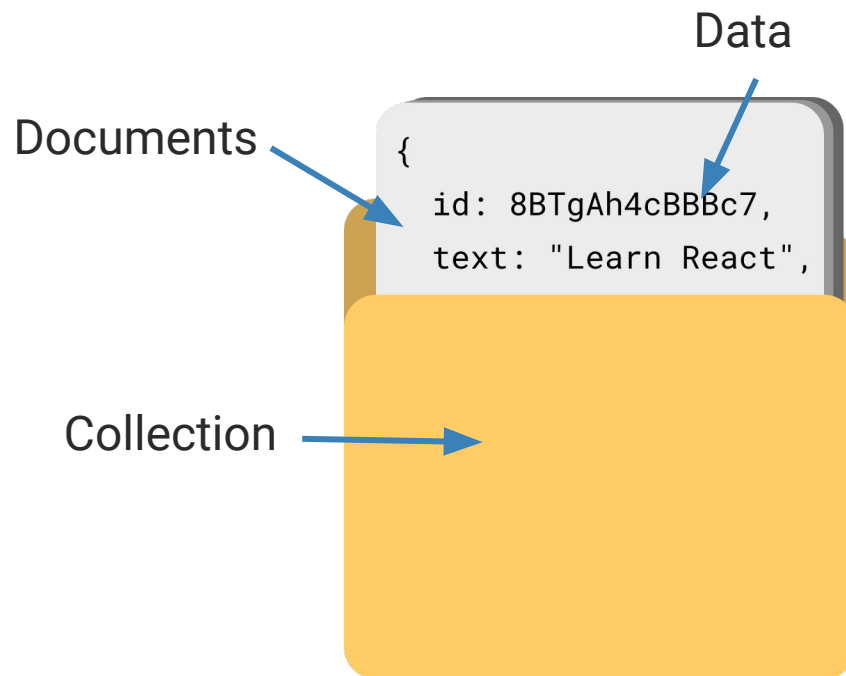


We recommend Cloud Firestore for most developers starting a new project. Cloud Firestore offers additional functionality, performance, and scalability on an infrastructure designed to support more powerful features in future releases. Expect to see new query types, more robust security rules, and improvements to performance among the advanced features planned for Cloud Firestore.



Datenmodell

- NoSQL, document-oriented database
- Datenbank besteht aus mehreren Collections mit Documents
- Document ist ein JSON-Objekt
- Document kann wiederum Collections beinhalten, etc.
- Vergleichbar mit MongoDB
 - 👉 Allerdings stark eingeschränkte Möglichkeiten für Queries (z.B. [keine Volltextsuche!](#))





Datenmodell

🏠 > todos > tbGRoTme3nwx...		
🔗 react-firebase-todo-app-ab936	📁 todos	📄 tbGRoTme3nwxXhxyHZfu
+ Add collection	+ Add document	+ Add collection
todos >	8BTgAh4cBBB1b710H0Zy	+ Add field
	tbGRoTme3nwxXhxyHZfu >	checked: true
		createdAt: November 1, 2018 at 1:52:52 PM UTC+1
		text: "Learn Firebase"
		userId: "thMnPv1UMhOMedyP9j0zstriojr1"



Daten erstellen und schreiben

- Auf Collections und Dokumente wird per *Referenz* zugegriffen:

```
const collectionRef = db.collection("todos")
```

```
const documentRef = db.collection("todos").doc("9bnh...")
```

- Über die **collectionRef** können Dokumente erstellt werden:

```
db.collection("todos").add({ text: "Learn Firebase" });
```

- Über die **documentRef** können Dokument bearbeitet werden:

```
db...doc("9bnh...").update({ text: "Learn Firestore" });
```



Daten abfragen

- Collection- und Dokument-Referenzen müssen nicht zwingend existieren



Abfragen sind immer asynchron und können auch fehlschlagen:

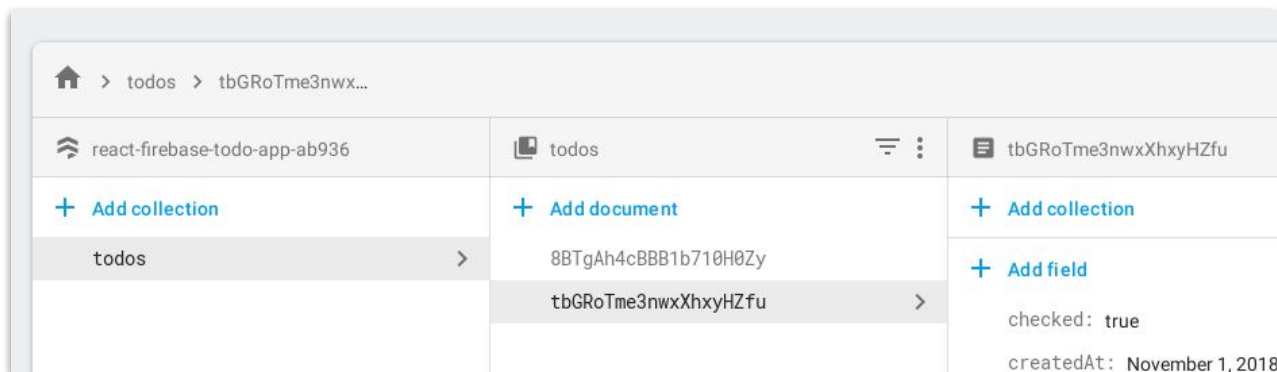
```
db.collection("todos").doc("just-some-id").get().then(doc => {  
  if (!doc.exists) {  
    console.log("No such document!");  
  } else {  
    console.log("Document data:", doc.data());  
  }  
}).catch(err => {  
  console.log("Error getting document", err);  
});
```




Daten abfragen

- Collections erlauben SQL-ähnliche Queries mit Filter und Sortierung

```
db.collection("todos").where("checked", "==", true).orderBy("createdAt")
  .get().then(snapshot => {
    snapshot.forEach(doc => {
      console.log(doc.id, "=>", doc.data());
    });
  })
.catch(err => ...);
```



- Bei Abfragen erhalten wir die **id** und **data** – den Dokumentinhalt – als Snapshot.



Realtime Updates

- Clients werden über Updates der Collections benachrichtigt:

```
db.collection("todos").where("userId", "==", auth.currentUser.uid)
  .onSnapshot(handleOnNext, handleOnError)
```

```
const handleOnNext = snapshot => {
  if (snapshot) {
    const data = snapshot.docs.map(doc => ({ id: doc.id, ...doc.data() }))
    ...
  }
};

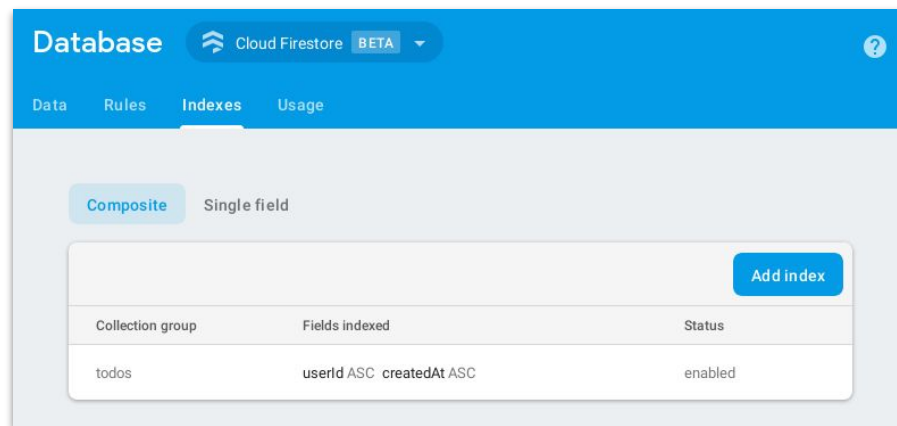
const handleOnError = error => {
};
```



Indexes

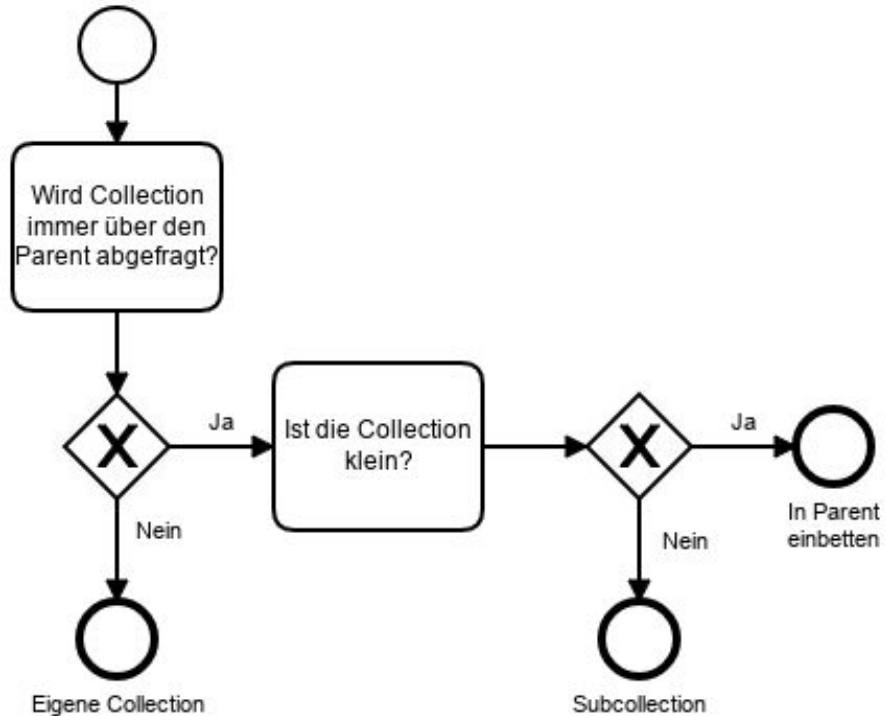
- Firebase erstellt automatische Indizes für **einzelne** Datenfelder
 - 🔥 Sortierung Ascending und Descending
 - 🔥 <, <=, ==, >=, und > Abfragen
- Erfolgt eine Abfrage über mehrere Felder, **muss** ein Index erstellt werden:

```
db.collection("todos")  
  .where("userId", "==", uid)  
  .orderBy("createdAt");
```



👉 Siehe auch firestore.indexes.json in der Übung.

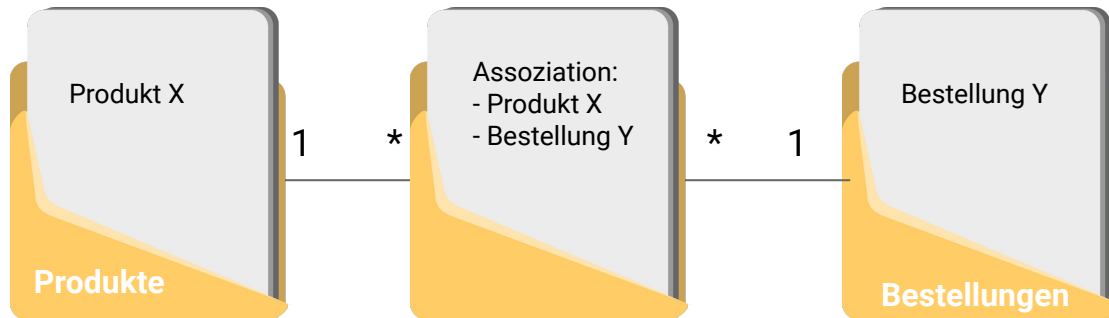
NoSQL: One-To-Many



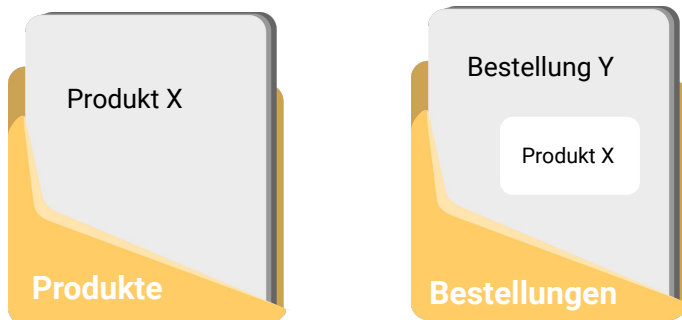
NoSQL: Many-To-Many



- Wie in relationaler Datenbank mit Assoziationstabelle



- Oder Daten kopieren und einbetten



NoSQL: Many-To-Many



- Assoziationstabelle vs. Einbetten
 - 👍 Kein Kopieren der Daten
 - 👎 Abfragen komplexer, da Firestore keine Joins unterstützt
- Kopieren der Daten muss kein Nachteil sein:
 - 👉 Preisänderung eines Produkts hat keinen Einfluss auf vergangene Bestellungen
 - 👉 Adressänderung eines Kunden verändert keine alten Bestellungen
 - 💪 Kopierte Daten können mittels Trigger und Cloud Function wieder synchronisiert werden

CAS FEE - Serverless Data Management

Firestore in React





Firebase Authentication

firebase / firebaseui-web-react

Watch

68

Star

261

Fork

46

Code

Issues 5

Pull requests 0

Projects 0

Wiki

Insights

React Wrapper for firebaseUI Web

102 commits

1 branch

12 releases

7 contributors

Apache-2.0

Branch: master

New pull request

Create new file

Upload files

Find file

Clone or download



nicolasgarnier dist v3.1.2

Latest commit 1701a0c 29 days ago

dist	dist v3.1.2	29 days ago
example	dist v3.1.2	29 days ago
src	Made className optional for flow	a month ago

- Library für die Authentication [firebaseui-web-react](#)
- Vorgefertigte Buttons für Login und Authentifizierungs-Flow

Konfiguration

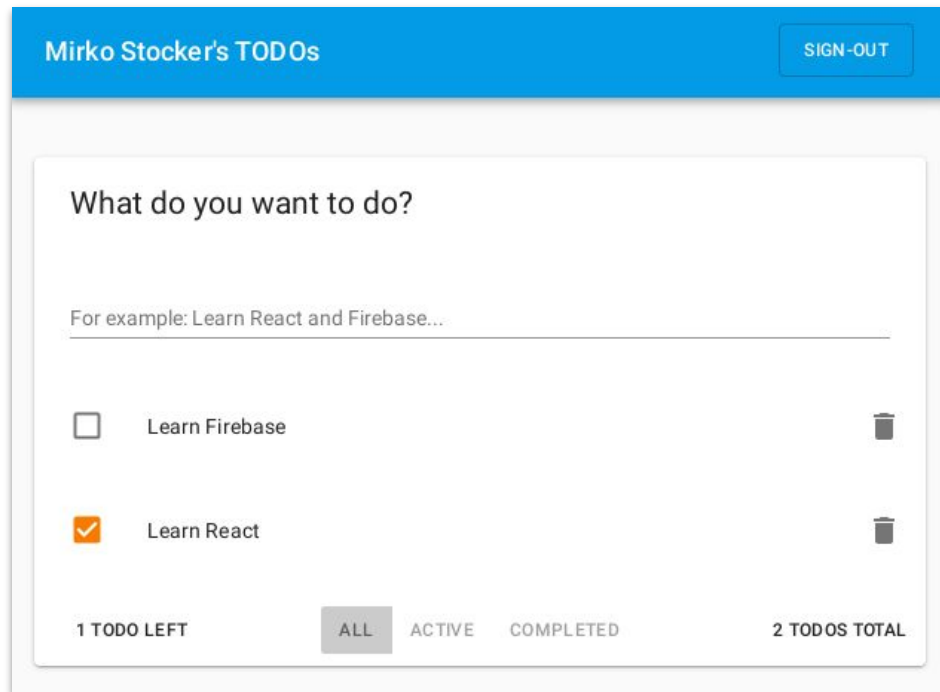
```
import firebase from "firebase/app";  
import "firebase/auth";  
import "firebase/firestore";  
  
import config from "./config";  
  
if (!firebase.apps.length) {  
  firebase.initializeApp(config);  
}  
  
...  
  
export const auth = firebase.auth();  
export const db = firebase.firestore();
```

config.js

```
export default {  
  apiKey: "...",  
  ...  
  projectId: "todo-app-ab936"  
};
```

Beispielanwendung

- Einfachste TODO App mit
 - 👉 [Material UI](#)
 - 👉 Firebase Authentifizierung
 - 👉 [Firebase Hosting](#)
 - 👉 TODOs erstellen, als erledigt markieren und löschen
 - 👉 Lokale Filterung der TODOs
 - 👉 Speicherung und Live-Synchronisation über alle Sessions mit Firestore



Weiterführende Links

- Authentifizierung für React-Router Routes

<https://www.robinwieruch.de/complete-firebase-authentication-react-tutorial/#react-firebase-protected-routes>

- React Firestore Library, ähnlich unserer TodosCollection

<https://github.com/green-arrow/react-firestore>

- Firebase mit React Hooks nutzen

<https://github.com/CSFrequency/react-firebase-hooks>

- User Ordered Data in Google Cloud Firestore

<https://dev.to/daviddalbusco/reorder-with-google-cloud-firestore-3dhl>

CAS FEE - Serverless Data Management

Danke für die Aufmerksamkeit

Mirko Stocker – mirko.stocker@ost.ch

